High Performance Object-Oriented Programming in Fortran 90

Viktor K. Decyk[1,2] and Charles D. Norton[2]

[1] Department of Physics and Astronomy
University of California at Los Angeles
Imaging and Spectrometry

[2] NASA Jet Propulsion Laboratory
High Performance Computing Systems Group
Systems Technology Section (385)

What is object-oriented programming?

The goal is to model software based on “real-world” abstractions enhancing safety, portability, modifiability, and understanding. This consists of analyzing the problem, designing objects, and writing software using a specific language.

Why Fortran 90?

Ambitious parallel applications impose new demands on software development. Fortran 90 support many new features beneficial for scientific programming. It is also a subset of High Performance Fortran and compatible with Fortran 77.

Fortran 90 features modernize programming.

Modules
Encapsulate data and routines across program units.

Derived Types
Allow user-defined types to be created.

Array Syntax
Simplifies operations on whole arrays, or array components.

Generic Interfaces
A single call can act differently based on the parameters.

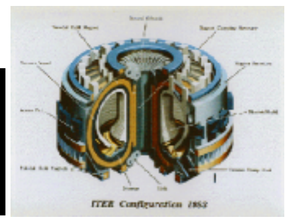
Use Association
Supports module interaction.

Pointers/Allocatable Arrays
Support the use of dynamic data structures.

Many additional intrinsics and features exist that redefine the nature of programming in Fortran!

How do Fortran 90 features benefit the development of complex simulation software (like Tokamak plasma modeling)?

Reprinted from PPPL web pages.



Features like modules, use association, derived types, array syntax, generic interfaces, overloading, and pointers simplify how problems are represented.

Many arguments to describe plasma features in Fortran 77

```
dimension part(idimp,npmax), q(nx,ny,nzpmx)
dimension fx(nx,ny,nzpmx),fy(nx,ny,nzpmx),fz(nx,ny,nzpmx)
data qme, dt /-1.,.2/
call push(part,fx,fy,fz,npp,qme,dt,wke,nx,ny,npmax,nzpmx)
call dpost(part,q,npp,noff,qme,nx,ny,npmax,nzpmx)
```

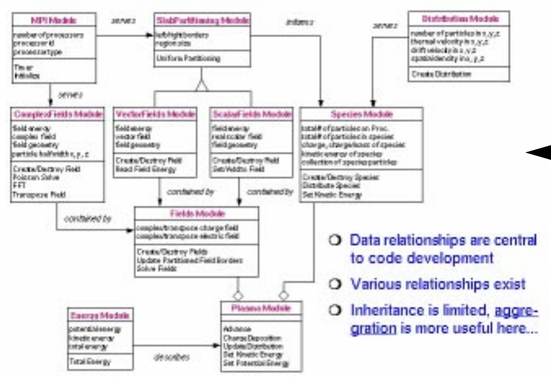
Fortran 90 allows encapsulation into logically related units

```
use partition_module, plasma_module
type (species) :: electrons
type (scalarfield) :: charge_density
type (vectorfield) :: efield
type (slabpartition) :: edges
real :: dt = .2
call plasma_push(electrons, efield, edges, dt)
call plasma_dpost(electrons, charge_density, edges)
```

Object-Oriented design models clarify application organization.

Relationships among data abstractions are the focus of programming, not procedure calls.

Fortran 90 object-oriented model.



How do Fortran 90 objects help here?

Object-Oriented techniques capture essential abstractions (particles, fields, relationships) allowing extension of codes to a variety of applications. This enhances collaborations and promotes better and safer software designs.

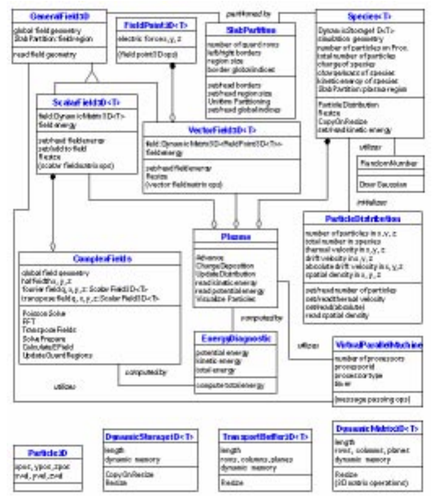
If the functionalilty is the same then why do these organizations look different?

The answer is due to language features. Fortran 90 supports multidimensional dynamic arrays, array operations, and other features that must be created explicitly in C++.

How does Fortran 90 support OO concepts?

Modules represent classes while variables, created from derived types in modules, represent objects. Module routines take objects as parameters.

C++ object-oriented model.



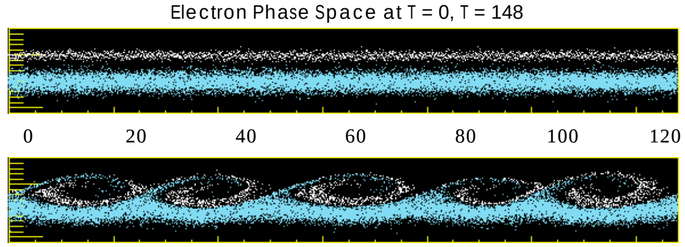
Application development on supercomputers.

Plasma particle-in-cell (PIC) codes have been developed with these techniques on the IBM SP and Cray T3E distributed-memory supercomputers using the General Concurrent PIC algorithm.

Beam-Plasma Instability Experiment

A moving particle beam is injected into a stationary background of higher density producing vortices, that are self-enhanced, in phase space.

Electron Phase Space at T = 0, T = 148

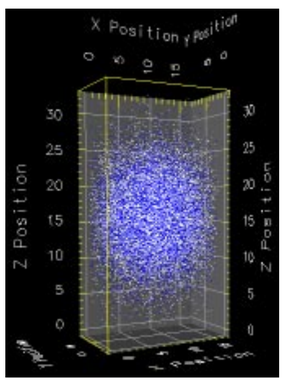


Sketch of object-oriented Fortran 90 parallel PIC code

```
PROGRAM beps3k
use partition_class, plasma_class
call MPI_INIT(ierror)
call species_init(electrons, qme, qbme, np)
call fields_init(cdensity, nx, ny, nz)
call fields_init(efield, nx, ny, nz)
do itime = 1,500
   call fields_solve(cdensity, efield)
   call plasma_getpe(energ, efield)
   call plasma_push(electrons, efield, edges, dt)
   call plasma_pmove(electrons, edges)
   call plasma_dpost(electrons, cdensity, edges)
end do
call fields_destroy(efield) ; call fields_destroy(cdensity)
call species_destroy(electrons)
call MPI_FINALIZE(ierror)
END PROGRAM beps3k
```

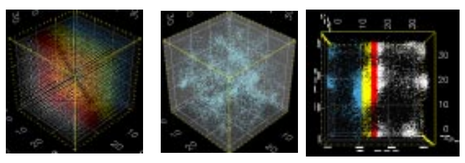
Free-Expansion Experiment

The electron-ion mixture expands from a dense region into vacuum under its own forces.



Gravitational Experiment

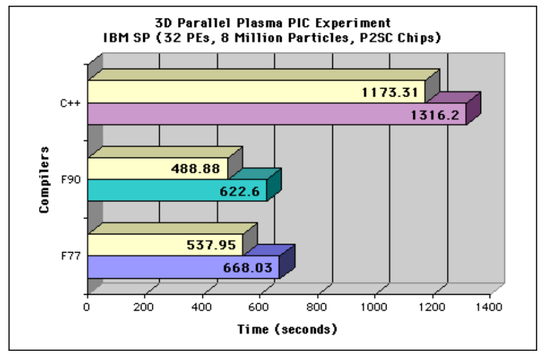
Initial partitioning of bodies with uniform distribution with subsequent clustering of bodies under gravitational forces.



How does Fortran 90 performance compare to C++ on supercomputers?

Fortran 90 typically exceeds Fortran 77 and C++ in such applications.

Here, the KAI C++ compiler, considered the most efficient, is compared to IBM’s compilers with the most aggressive optimizations for a 3D parallel PIC code. The C++ compilation lasted two hours compared to 5 minutes for Fortran 90. (Yellow bars indicate P2SC hardware optimizations.)



Does Fortran 90 support interlanguage communication of objects with C++?

Yes, as long as C++ virtual functions are not used since this construct modifies the object data storage layout in a vendor-dependent manner. (The F90 sequence attribute is suggested.)

C++ and Fortran 90 inheritance can also be used and member routines can be called.

Are all object-oriented features useful in scientific programming?

Not all features of object-oriented design seem useful for scientific programming. Inheritance has limited usefulness while compostion of abstractions were very useful.

The major benefit...

Fortran 90 creates new opportunities to attack more ambitious problems in a manageable way.

What Lies Ahead?

Fortran 2000 will have full object-oriented features including single inheritance, polymorphic objects, parameterized derived types, constructors, and destructors.

See for yourself!

Visit our web sites for more information:
http://www.cs.rpi.edu/~szymansk/oof90.html
http://www-hpc.jpl.nasa.gov

Or write to us:
decyk@physics.ucla.edu
Charles.D.Norton@jpl.nasa.gov

Created:
09 / 05 / 97